# BOUNDARY HANDLING FOR CONVOLUTIONAL SPARSE REPRESENTATIONS

*Brendt Wohlberg*

Theoretical Division
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

## ABSTRACT

Convolutional sparse representations differ from the standard form in representing the signal to be decomposed as the sum of a set of convolutions with dictionary filters instead of a linear combination of dictionary vectors. The advantage of the convolutional form is that it provides a single-valued representation optimised over an entire signal. The substantial computational cost of the convolutional sparse coding and dictionary learning problems has recently been shown to be greatly reduced by solving in the frequency domain, but the periodic boundary conditions imposed by this approach have the potential to create boundary artifacts. The present paper compares different approaches to avoiding these effects in both sparse coding and dictionary learning.

***Index Terms—*** Convolutional Sparse Coding, Convolutional Dictionary Learning, Boundary Effects

## 1. INTRODUCTION

Of the variety of formulations of the sparse coding problem [1, 2], the present paper considers the specific form of Basis Pursuit De-Noising (BPDN) [3]

$$\arg\min_{\mathbf{x}} \frac{1}{2} \left\|D\mathbf{x} - \mathbf{s}\right\|_2^2 + \lambda \left\|\mathbf{x}\right\|_1 \;. \tag{1}$$

A convolutional sparse representation [4] replaces the representation as a linear combination of dictionary matrix columns with a sum of a set of convolutions with dictionary filters. The convolutional variant of BPDN, which will be referred to here as Convolutional BPDN (CBPDN) is defined as

$$\arg\min_{\{\mathbf{x}_m\}} \frac{1}{2} \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big\|_2^2 + \lambda \sum_m \left\|\mathbf{x}_m\right\|_1 \;, \tag{2}$$

where $\{\mathbf{d}_m\}$ is a set of $M$ dictionary *filters*, $*$ denotes convolution, and $\{\mathbf{x}_m\}$ is a set of coefficient maps[1].

Problem (2) can be very computationally expensive when $N$ and $M$ are not small. The authors of [4] observed that a frequency domain solution was possible, but claimed that a spatial domain approach was to be preferred due to the possibility of boundary artifacts arising from the periodic boundary conditions inherent in the use of the Discrete Fourier Transform (DFT). However, a number of authors have recently proposed much more efficient algorithms based on the Alternating Direction Method of Multipliers (ADMM) [5] framework, solving the most computationally expensive of the resulting sub-problems in the DFT domain [6, 7, 8, 9]. Bristow et al. [6] provided some experimental evidence that the practical impact of boundary artifacts on dictionary learning is negligible, but Heide et al. [9] have suggested that this is not always the case, and proposed a new approach designed to avoid boundary effects. The present paper compares the boundary handling approach of Heide et al. [9] with alternatives, including an existing method that has, thus far, only been very briefly described in the literature [10, Sec. 4].

This research was supported by the U.S. Department of Energy via the LANL/LDRD Program, and by UC Lab Fees Research grant 12-LR-236660.

[1]For notational simplicity $\mathbf{s}$ and each of the $\{\mathbf{x}_m\}$ are considered to be $N$ dimensional vectors, where $N$ is the number of pixels in an image.

## 2. ADMM ALGORITHM FOR CBPDN

We start by describing the common form of the ADMM algorithms with periodic boundary conditions [6, 7, 8]. First, define $D_m$ is a linear operator such that $D_m \mathbf{x}_m = \mathbf{d}_m * \mathbf{x}_m$, and define block matrices and vectors

$$D = \begin{pmatrix} D_0 & D_1 & \dots \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \end{pmatrix} \;, \tag{3}$$

so that (2) can be written in the same form as (1). The simplest variable splitting into the ADMM standard form is

$$\arg\min_{\mathbf{x},\mathbf{y}} \frac{1}{2} \left\|D\mathbf{x} - \mathbf{s}\right\|_2^2 + \lambda \left\|\mathbf{y}\right\|_1 \;\text{ s.t. } \mathbf{x} - \mathbf{y} = 0 \;, \tag{4}$$

for which the corresponding ADMM iterations are

$$\mathbf{x}^{(j+1)} = \arg\min_{\mathbf{x}} \frac{1}{2} \left\|D\mathbf{x} - \mathbf{s}\right\|_2^2 + \frac{\rho}{2} \left\|\mathbf{x} - \mathbf{y}^{(j)} + \mathbf{u}^{(j)}\right\|_2^2 \tag{5}$$

$$\mathbf{y}^{(j+1)} = \arg\min_{\mathbf{y}} \lambda \left\|\mathbf{y}\right\|_1 + \frac{\rho}{2} \left\|\mathbf{x}^{(j+1)} - \mathbf{y} + \mathbf{u}^{(j)}\right\|_2^2 \tag{6}$$

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + \mathbf{x}^{(j+1)} - \mathbf{y}^{(j+1)} \;. \tag{7}$$

The solution to (6) is given by

$$\mathbf{y}_m^{(j+1)} = \mathcal{S}_{\lambda/\rho}\Big(\mathbf{x}_m^{(j+1)} + \mathbf{u}_m^{(j)}\Big) \;, \tag{8}$$

where

$$\mathcal{S}_\gamma(\mathbf{u}) = \operatorname{sign}(\mathbf{u}) \odot \max(0, |\mathbf{u}| - \gamma) \tag{9}$$

is the well-known shrinkage/soft thresholding operation. The only computationally expensive step is (5), which the DFT convolution theorem implies is equivalent to the DFT domain problem

$$\arg\min_{\hat{\mathbf{x}}} \frac{1}{2} \left\|\hat{D}\hat{\mathbf{x}} - \hat{\mathbf{s}}\right\|_2^2 + \frac{\rho}{2} \left\|\hat{\mathbf{x}} - (\hat{\mathbf{y}} - \hat{\mathbf{u}})\right\|_2^2 \;, \tag{10}$$

where $\hat{D} = \begin{pmatrix} \hat{D}_0 & \hat{D}_1 & \dots \end{pmatrix}$ and

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \\ \vdots \end{pmatrix} \quad \hat{\mathbf{y}} = \begin{pmatrix} \hat{\mathbf{y}}_0 \\ \hat{\mathbf{y}}_1 \\ \vdots \end{pmatrix} \quad \hat{\mathbf{u}} = \begin{pmatrix} \hat{\mathbf{u}}_0 \\ \hat{\mathbf{u}}_1 \\ \vdots \end{pmatrix} \;, \tag{11}$$

with $\hat{\mathbf{z}}$ denoting the DFT of variable $\mathbf{z}$. The solution for (10) is given by the linear system

$$(\hat{D}^H \hat{D} + \rho I)\hat{\mathbf{x}} = \hat{D}^H \hat{\mathbf{s}} + \rho\,(\hat{\mathbf{y}} - \hat{\mathbf{u}}) \;, \tag{12}$$

which can be solved very efficiently by exploiting the Sherman-Morrison formula [7, 8].

## 3. BOUNDARY HANDLING

The approach outlined in the previous section applies convolution in the DFT domain, implictly imposing periodic boundary conditions, which can be expected to result in boundary artifacts when representing signals that are not circularly symmetric.

### 3.1. Boundary Overlap Suppression

The primary concern associated with the use of periodic boundary conditions in a convolutional sparse representation is that the representation can include filters that straddle the signal boundary. Since the discontinuities in these regions will not, in general, conform to the signal model for which the dictionary was learned, the representation can be expected to be locally inferior.

A simple solution is to modify (2) to include the constraint that $B\mathbf{x}_m = 0 \;\forall m$, where $B$ is an operator projecting each $\mathbf{x}_m$ to the subset of elements in that map that corresponds to filters with support extending across the image boundary. The indicator function[2] can be used to rewrite the constrained problem in unconstrained form

$$\mathop{\arg\min}_{\{\mathbf{x}_m\}} \frac{1}{2}\left\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\right\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 + \sum_m \iota_C(\mathbf{x}_m) \;, \tag{13}$$

with $C = \{\mathbf{x} \in \mathbb{R}^N : B\mathbf{x} = 0\}$. Since the proximal map of the sum of the $\ell^1$ and indicator function terms can be computed in closed form, it is not necessary to introduce an additional variable splitting to deal with the indicator function term, the only modification to ADMM iterations (5)–(7) being that the $\{\mathbf{y}_m\}$ update becomes

$$\mathbf{y}_m^{(j+1)} = B^T B\, \mathcal{S}_{\lambda/\rho}\left(\mathbf{x}_m^{(j+1)} + \mathbf{u}_m^{(j)}\right) \;, \tag{14}$$

i.e. shrinkage followed by projection onto the feasible set $C$. This method will be referred to as boundary overlap suppression (BOS).

This approach allows for convolution in the DFT domain that is equivalent to a spatial domain convolution that excludes any filter position that overlaps the signal boundary, circumventing wrap-around artifacts from the circular convolution. However, while artifacts resulting from filters crossing the boundary are avoided, a different type of artifact may result from the pixels near the boundary being represented by fewer spatial shifts of the dictionary filters than the pixels in the interior of the image, so that the boundary regions have at their disposal a reduced set of possible representations. This approach is of interest, however, since it corresponds to the most common boundary handling for standard patch-based sparse representations. The usual boundary handling for convolutional sparse representations computed in the spatial domain also has the potential for an inferior representation near the boundary for similar reasons [11, Sec. 2.1],[6].

[2]The indicator function of a set $S$ is defined as

$$\iota_S(X) = \begin{cases} 0 & \text{if } X \in S \\ \infty & \text{if } X \notin S \end{cases} \;.$$

### 3.2. Boundary Masking

To avoid any boundary artifacts, it is necessary to allow the filter kernels to overlap the boundary, but without wrap-around, equivalent to allowing partial patches on the boundary region within a patch-based scheme. This can be achieved by using coefficient maps $\{\mathbf{x}_m\}$ and signal $\mathbf{s}$ that have been spatially extended by zero-padding, and solving

$$\mathop{\arg\min}_{\mathbf{x}} \frac{1}{2}\left\|WD\mathbf{x} - \mathbf{s}\right\|_2^2 + \lambda \|\mathbf{x}\|_1 \;. \tag{15}$$

where $W$ is a spatial mask operator that zeros-out any regions of its argument that extend beyond the original boundary, and $D$ and $\mathbf{x}$ are as defined in (3). Direct solution of this problem in the DFT domain is not possible, however, since a spatial mask does not have a compact representation in the DFT domain.

## 4. EFFICIENT MASKING ALGORITHMS

We consider two different approaches for efficient DFT domain solution of a problem of the form of (15).

### 4.1. Additive Mask Simulation

The fundamental idea of the first approach is to introduce into the representation an additive component that is constrained to be zero within the part of the signal retained by the mask (so that it does not perturb the retained part of the representation), and is unconstrained and un-penalized within the masked-out region (so that it takes on values that cancel any influence on the functional value from parts of filters that protrude into the masked-out region) [10]. This additional component can be introduced into the standard form of the problem by adding an impulse filter to the dictionary; the corresponding coefficient map becomes the additional component, and it is just necessary to exclude this map from the $\ell^1$ norm and apply the additional constraint that it is zero where the mask value is unity. This method will be referred to as additive mask simulation (AMS).

Formally, given a dictionary with $M$ filters $\{\mathbf{d}_m\}_{m \in \{0,1,\ldots,M-1\}}$, append impulse filter $\mathbf{d}_M = \boldsymbol{\delta}$ to give an extended dictionary with $M+1$ filters. The mask is represented as a diagonal weighting matrix, $W$, with entries that are zeros and ones. The problem can be written as

$$\mathop{\arg\min}_{\{\mathbf{x}_m\}} \frac{1}{2}\left\|\sum_{m=0}^{M} \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\right\|_2^2 + \lambda \sum_{m=0}^{M-1} \|\mathbf{x}_m\|_1 + \iota_C(\mathbf{x}_M) \;, \tag{16}$$

where $\iota_C(\cdot)$ is the indicator function of the set $C = \{\mathbf{x} \in \mathbb{R}^N : W\mathbf{x} = 0\}$. This problem has the same form as (13), and can be solved by a minor variation on the algorithm for that problem.

### 4.2. Mask Decoupling

More recently, Heide et al. [9] have proposed applying the mask decoupling technique (abbreviated as MD when convenient) of Almeida and Figueiredo [12] to the masked CBPDN problem (15). (In contrast to the AMS method described above, this method does not require that the mask have binary (i.e. 0 or 1) entries.) Instead of the usual splitting into ADMM standard form as in (4), this technique employs the splitting

$$\begin{aligned} &\mathop{\arg\min}_{\mathbf{x},\mathbf{y}_0,\mathbf{y}_1} \frac{1}{2}\left\|W\mathbf{y}_1 - \mathbf{s}\right\|_2^2 + \lambda \|\mathbf{y}_0\|_1 \\ &\text{s.t.} \begin{pmatrix} \mathbf{x} \\ D\mathbf{x} \end{pmatrix} - \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{pmatrix} = 0 \;. \end{aligned} \tag{17}$$

Defining

$$A = \begin{pmatrix} I \\ D \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{pmatrix} \quad \mathbf{u} = \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix} , \tag{18}$$

the corresponding ADMM iterations are

$$\mathbf{x}^{(j+1)} = \operatorname*{arg\,min}_{\mathbf{x}} \frac{\rho}{2} \left\| A\mathbf{x} - \mathbf{y}^{(j)} + \mathbf{u}^{(j)} \right\|_2^2 \tag{19}$$

$$\mathbf{y}^{(j+1)} = \operatorname*{arg\,min}_{\mathbf{y}} \frac{1}{2} \left\| W\mathbf{y}_1 - \mathbf{s} \right\|_2^2 + \lambda \left\| \mathbf{y}_0 \right\|_1 + \frac{\rho}{2} \left\| A\mathbf{x}^{(j+1)} - \mathbf{y} + \mathbf{u}^{(j)} \right\|_2^2 \tag{20}$$

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + A\mathbf{x}^{(j+1)} - \mathbf{y}^{(j+1)} . \tag{21}$$

The functional minimised in (19) can be expanded as

$$\frac{\rho}{2} \left\| A\mathbf{x} - \mathbf{y} + \mathbf{u} \right\|_2^2 = \frac{\rho}{2} \left\| \begin{pmatrix} \mathbf{x} \\ D\mathbf{x} \end{pmatrix} - \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{pmatrix} + \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix} \right\|_2^2 \tag{22}$$

$$= \frac{\rho}{2} \left\| D\mathbf{x} - (\mathbf{y}_1 - \mathbf{u}_1) \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{x} - (\mathbf{y}_0 - \mathbf{u}_0) \right\|_2^2 , \tag{23}$$

which is of the same form as (5), and can be solved via the same frequency domain method. The functional minimised in (20) can be expanded as

$$\frac{1}{2} \left\| W\mathbf{y}_1 - \mathbf{s} \right\|_2^2 + \lambda \left\| \mathbf{y}_0 \right\|_1 + \frac{\rho}{2} \left\| \mathbf{y}_1 - (D\mathbf{x} + \mathbf{u}_1) \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{y}_0 - (\mathbf{x} + \mathbf{u}_0) \right\|_2^2 . \tag{24}$$

Since the $\mathbf{y}_0$ and $\mathbf{y}_1$ components of $\mathbf{y}$ are decoupled, minimisation with respect to $\mathbf{y}$ can be achieved by the independent minimisations

$$\mathbf{y}_0^{(j+1)} = \operatorname*{arg\,min}_{\mathbf{y}_0} \lambda \left\| \mathbf{y}_0 \right\|_1 + \frac{\rho}{2} \left\| \mathbf{y}_0 - (\mathbf{x} + \mathbf{u}_0) \right\|_2^2 \tag{25}$$

$$\mathbf{y}_1^{(j+1)} = \operatorname*{arg\,min}_{\mathbf{y}_1} \frac{1}{2} \left\| W\mathbf{y}_1 - \mathbf{s} \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{y}_1 - (D\mathbf{x} + \mathbf{u}_1) \right\|_2^2 . \tag{26}$$

The solution to (25) is just soft thresholding, and the solution to (26) is given by

$$(W^T W + \rho I)\mathbf{y}_1 = W^T \mathbf{s} + \rho(D\mathbf{x} + \mathbf{u}_1) . \tag{27}$$

## 5. DICTIONARY LEARNING

Dictionary learning based on CBPDN can be expressed as

$$\operatorname*{arg\,min}_{\mathbf{x},D} \frac{1}{2} \left\| D\mathbf{x} - \mathbf{s} \right\|_2^2 + \lambda \left\| \mathbf{x} \right\|_1 , \tag{28}$$

where $D$ and $\mathbf{x}$ are as defined in (3), and the problem is solved by alternating between minimizing with respect to $\mathbf{x}$ and $D$. The minimization with respect to $\mathbf{x}$ can be achieved using any of the sparse coding variants described in the preceding sections, and the minimization with respect to $D$ can also be solved via an ADMM algorithm [8, Sec. V]). If the sparse coding variant supports masking, so does the resulting dictionary learning algorithm. Some additional care is required in these cases (e.g., when the sparse coding utilises additive mask simulation, the dictionary update step should exclude the impulse component of the dictionary), but the additional complications are implementational rather than conceptual.

A much more serious issue is the question of how the sparse coding and dictionary update stages are to be combined to to construct a dictionary learning algorithm. Heide et al. [9] follow the example of Bristow et al. [6] in deriving the dictionary learning algorithm from a single Augmented Lagrangian [5] functional (although with multiple consecutive steps of each sparse coding and dictionary update stage instead of interleaving a single step of each stage). This derivation requires that sparse coding and dictionary update stages are interleaved on their primary variables (e.g. variable $\mathbf{x}$ in the sparse coding algorithm represents the coefficient maps in the dictionary update stage). The approach advocated here, in contrast, follows that proposed in [8, Sec. V.B] in which the sparse coding and dictionary update stages are interleaved on their auxiliary variables (e.g. variable $\mathbf{y}$ in the sparse coding algorithm represents the coefficient maps in the dictionary update stage), in a way that is not derivable from a single Augmented Lagrangian functional.

To avoid convoluted descriptions, the dictionary learning algorithm of Heide et al. [9] will be referred to as primary variable alternation with mask decoupling (PVA-MD), and the structure advocated here will be referred to as auxiliary variable alternation (AVA), with a postfix indicating a specific sparse coding variant (e.g. AVA-Std for AVA with standard CBPDN, or AVA-AMS for AVA with additive mask simulation) when appropriate.

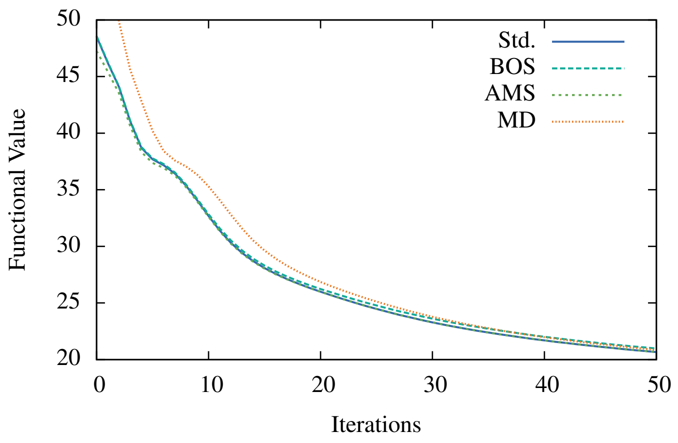

**Fig. 1**. A comparison of functional value decay for standard CBPDN and BOS, AMS, and MD variants, with $\lambda = 0.01$. Note that the BOS method has a slightly larger minimum functional value than the other three methods, which becomes apparent at a larger number of iterations than plotted here.

## 6. RESULTS

We now compare the performance of the different boundary handling methods in both sparse coding and dictionary learning problems.

### 6.1. Sparse Coding

The performance of the previously-discussed sparse coding variants was compared in three different sets of experiments. For all three sets the dictionary consisted of 108 filters of size $12 \times 12$, and the test image was the $512 \times 512$ pixel "Lena" image after scaling of pixel values to the range $[0, 1]$ and highpass filtering, and a grid search was performed to ensure a good choice of the penalty parameter $\rho$.

The first experiment compared the decay with number of iterations and time of the functional value for the standard CPBDN and BOS, AMS, and MD variants on an image without boundary exten-

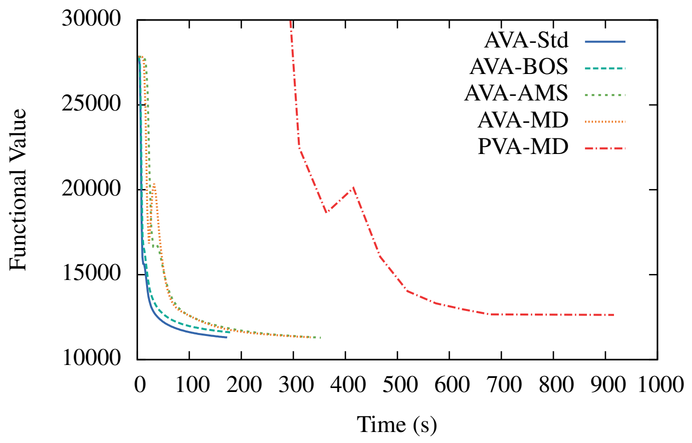

**Fig. 2**. A comparison on a set of 10 images of size $100 \times 100$ pixels of functional value decay for auxiliary variable alternation (AVA) dictionary learning with no boundary handling (Std), and with the BOS, AMS, and MD boundary handling methods, as well as with primary variable alternation with mask decoupling (PVA-MD).

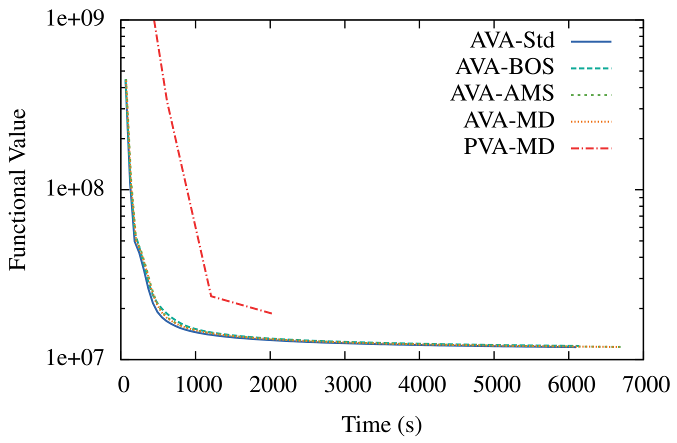

**Fig. 3**. A comparison on a set of 10 images of size $512 \times 512$ pixels of functional value decay for auxiliary variable alternation (AVA) dictionary learning with no boundary handling (Std), and with the BOS, AMS, and MD boundary handling methods, as well as with primary variable alternation with mask decoupling (PVA-MD).

sion and a constant unit mask (i.e. no masked-out regions). As can be seen in Fig. 1, the standard, BOS, and AMS methods have very similar convergence, with the MD method slightly behind at first, but rapidly approaching the behaviour of the other two methods.

The second experiment compared the decay with number of iterations and time of the functional value for the AMS and MD variants of CBPDN on the test image with a padding of 11 pixels on the bottom and right boundaries and a mask with zero entries corresponding to the extended region, and the third experiment compared the performance of the two mask implementation methods in a random "inpainting" problem, with the mask corresponding to randomly distributed corrupted pixels instead of a boundary region. The results of these experiments (omitted due to space constraints) were in agreement with those of the first in that the convergence of the AMS method is slightly faster than that of the MD method, the difference shrinking with increasing iteration number.

### 6.2. Dictionary Learning

Both sets of dictionary learning experiments presented here compare the AVA dictionary learning structure, with standard CBPDN and the BOS, AMS, and MD variants used for the sparse coding stage (using modified versions of Matlab implementations from the SPORCO library [13]) with the PVA-MD dictionary learning algorithm proposed by Heide et al. [9] (using their publicly available Matlab implementation [14]).

The first set of dictionary learning experiments compared a number of different approaches in the dictionary learning test reported by Heide et al. [9]: learning a dictionary of 100 filters of size $11 \times 11$, with $\lambda = 1$, on 10 images, of size $100 \times 100$ pixels and with values in the range $[0, 255]$, from the "fruits" dataset. The results were generated using 100 iterations of the AVA algorithms, and 13 outer iterations of the PVA-MD algorithm, which corresponds to 130 iterations of the AVA methods since it uses 10 inner iterations per outer iteration. It can be seen from Fig. 2 that (i) the standard CPBDN and BOS variants of AVA are approximately twice as fast as the AMS and MD variants (but note that the BOS variant has a slighly larger minimum functional value than the other methods), and (ii) the AVA-AMS and AVA-MD algorithms have very similar performance, and have much faster convergence than the PVA-MD algorithm.

The second set of experiments was similar to the first, but with the "fruits" data set replaced with a set of 10 greyscale images of size $512 \times 512$ pixels cropped from a set of images with a Creative Commons license on Flickr. The results were generated using 100 iterations of the AVA algorithms, and 3 outer iterations[3] of the PVA-MD algorithm, equivalent to 30 iterations of the other algorithms. It can be seen from Fig. 3 that (i) PVA-MD has far slower convergence than the other methods, which all have similar behaviour on the scale of this graph, and (ii) for the larger training images, the standard CPBDN and BOS AVA variants are not much faster than the AVA-AMS and AVA-MD variants. It should also be noted that the PVA-MD algorithm has far greater memory requirements than the other approaches, presumably due to the strategy of caching Cholesky factorizations of system matrices [9, Sec. 3]. For example, for this set of experiments, the maximum memory usage of the AVA-MD dictionary learning was 14GB, while that for PVA-MD was 73GB.

## 7. CONCLUSIONS

The BOS method has negligible additional computational cost over standard CBPDN, but is not expected to be as effective as boundary masking in addressing boundary artifacts[4]. The two different boundary masking algorithms have very similar computational performance, with a small advantage to the AMS method in all of the sparse coding comparisons presented here, and no overall advantage to the AVA forms of either method in the dictionary learning comparisons (AVA-AMS converges very slightly faster than AVA-MD in Fig. 2, and vice versa in Fig. 3). For dictionary learning, a far more significant issue is the way in which the sparse code and dictionary updates are interleaved, the MD variant of the AVA structure proposed in [8, Sec. V]) having much faster convergence and lower memory requirements than the PVA-MD structure proposed in [9].

[3]It was not possible to coerce the code [14] to continue beyond three outer iterations by any adjustment of the iteration limit and tolerance parameters.

[4]A comparison of the relative effectiveness of these methods in suppressing boundary artifacts is beyond the scope of the present paper, which focusses on computational issue, and will be addressed in future work.

## 8. REFERENCES

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009. doi:10.1137/060657704

[2] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014. doi:10.1561/0600000058

[3] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998. doi:10.1137/S1064827596304010

[4] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, Jun. 2010, pp. 2528–2535. doi:10.1109/cvpr.2010.5539957

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010. doi:10.1561/2200000016

[6] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, Jun. 2013, pp. 391–398. doi:10.1109/CVPR.2013.57

[7] B. Wohlberg, "Efficient convolutional sparse coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2014, pp. 7173–7177. doi:10.1109/ICASSP.2014.6854992

[8] ——, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016. doi:10.1109/TIP.2015.2495260

[9] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, 2015, pp. 5135–5143. doi:10.1109/CVPR.2015.7299149

[10] B. Wohlberg, "Endogenous convolutional sparse representations for translation invariant image subspace models," in *Proc. IEEE Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 2859–2863. doi:10.1109/ICIP.2014.7025578

[11] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. A. LeCun, "Learning convolutional feature hierachies for visual recognition," in *Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1090–1098.

[12] M. S. C. Almeida and M. A. T. Figueiredo, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3074–3086, Aug. 2013. doi:10.1109/tip.2013.2258354

[13] B. Wohlberg, "SParse Optimization Research COde (SPORCO)," Matlab library available from http://math.lanl.gov/~brendt/Software/SPORCO/, 2015, version 0.0.3.

[14] F. Heide, W. Heidrich, and G. Wetzstein, "Code and datasets for *Fast and Flexible Convolutional Sparse Coding*," Available from http://www.cs.ubc.ca/labs/imager/tr/2015/FastFlexibleCSC/#files, 2015.